Rebuilding a Plane In Flight

Refactors Under Pressure

Susan Sons
IU-CACR, OSG
sesons@iu.edu

http://hdl.handle.net/2022/21719

Disasters Happen.

An ounce of prevention is worth a pound of cure...

...but sometimes a pound, or a ton, of cure is needed.

We cannot go back in time. Facts are, we live in an imperfect world and some people will need to clean up messes: messes of their own making, and of others'.

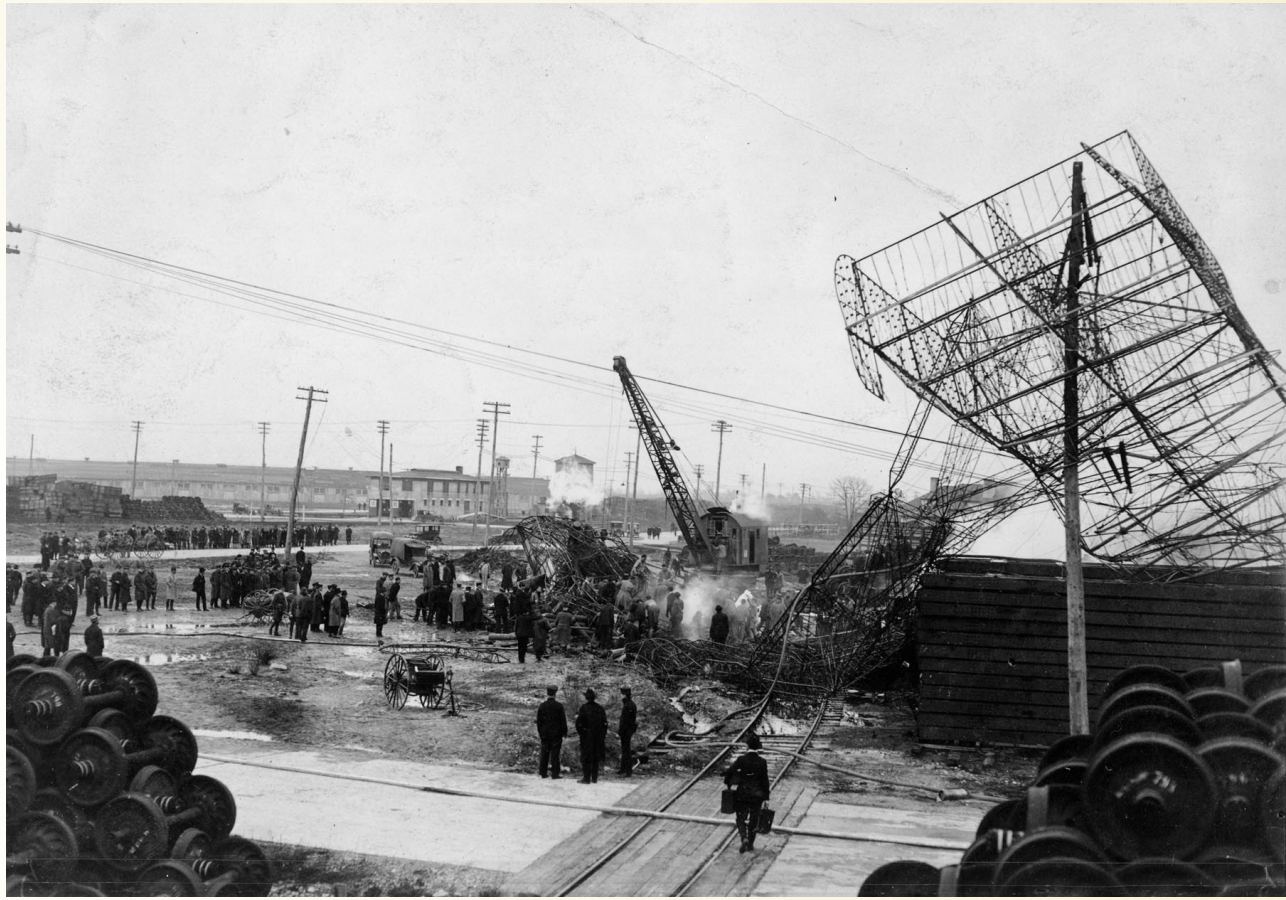


Image:

http://www.nationalmuseum.af.mil/Visit/Museum-Exhibits/Fact-Sheets/Display/Article/196943/roma-tragedy/

Working a critical disaster, joys and challenges

- The work is important.
- The bar for success is known.
- Success requires managing as many social problems as technical ones.
- The ground will change beneath you.
- There's pleasure in taking on the things that others are afraid to.

Why get good at this?

What does it take to be good at this?

Step Zero: Decide that you will be responsible.

- Understand the problem: this requires technical acumen, social skills, and domain knowledge. If you aren't yourself an experienced architect in this domain, you must have a very close partnership with one.
- Set a clear, concrete, finite scope.
- Spend time with people -- split technical and social leadership positions if needed to make this investment possible.
- **Expect drama. Forgive drama.**
- **Keep perspective: the purpose of a rescue is long-term sustainability. Any other goal may be sacrificed to support this one.**

Systems and Software

One Method

How to Train for the Unknown

Breaking it down: Priorities

Three themes must be considered at each step in planning and carrying out the refactor:

- **The Timeline Balancing Act:**
  Playing the long game while dealing with immediately pressing concerns, keeping perspective
- **Project Management Concerns:**
  Resourcing, communication, stakeholder priorities
- **Technical and Architectural Strategy:**
  Supporting toolchains, architecture principles, testing, code cleanliness, maintainability, security

Your refactor's most precious and finite resource is TIME.

- **Go for the snowball effect.**
- **Cluster disruptions in order to minimize them.**
- **Avoid Mythical Man-Month errors.**
- **Stay out of rabbit holes.**
- **Put long-term gains ahead of immediately pleasing people.**

Q & A

Project Management:

You will not know the depth, breadth, or nature of the social and technical problems until you are halfway down into the abyss.

There is always another problem lurking.

That's okay.

Code Longevity:

- Resources
- Personnel (devpower & expertise)
- Repository & Access
- Build System
- Tests
- Documentation
- Communication Channels

Pony Factor

How many *currently active* committers account for >50% of the code base?

$$\sum_{n=1}^{P} C_n \geq K \cdot V$$

Breakdown by Dave Nalley:

https://ke4qqq.wordpress.com/2015/02/08/pony-factor-math/

Based on research by Daniel Gruno of Snoot.io

Start With Recon and Comms

When Sputnik crashes down on your head, resist the urge to react immediately, unless it's to prevent immediate loss of life. Gather information, start identifying the problem and scoping a response, and talk to people.

Write. Write down your background planning, your thinking, your project scope. Then, communicate with stakeholders face-to-face (or by teleconference) and follow up in writing.

Be kinder to everyone than you need to be, be empathetic even when people are being wrong. Not because you're a sap, because it's how you get people to do communicate freely. Every disaster got that way somehow, and everyone near it fears blame. Leading a major refactor/rescue means keeping your focus more on outcomes than blame.

DO NOT try to plan a smooth-running project.

You must plan for drama and messiness so that you are able to absorb it.

Give yourself--and your team--healthy margins for error. This is how you beat code that is full of unknown landmines.

Stakeholders don't care how hard your job is.

- It's your job to find the stakeholders; they won't come find you.
- What are they trying to do with the software or system? What constraints do they operate under?
- It's also your job to sort out the XY problems.
- Manage expectations, and minimize negative impact on stakeholders.

It's under control. I have a process for this.

Even though I'm mostly (completely) winging it.

A complex refactor requires a team.

An effective refactor team is a group of humans who:

- Have complementary skill sets, and a diversity of outlooks.
- In aggregate, have all of the skills needed to complete the refactor.
- Have or can quickly build a working rapport that allows for comfortable, informal conversation.
- Have bought in to the refactor process.
- Have enough resources to do the work that's needed.
- Can check some ego at the door.

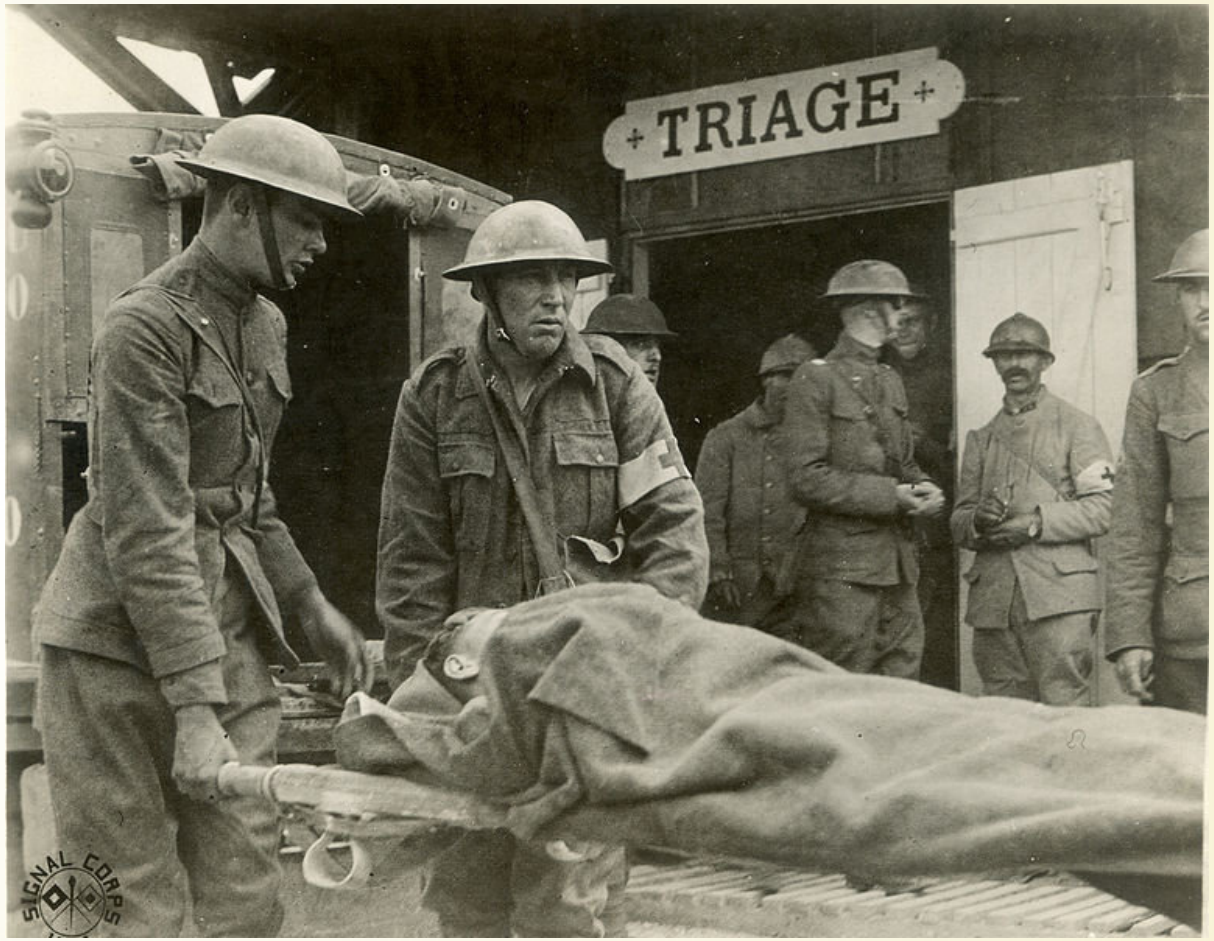
Q & A

So…this technology thing…

Triage

Triage is not about understanding the situation in total.

Triage is discovering the greatest points of crisis and how they relate to one another, so that the patient can be stabilized to the point that we can worry about their general health.




Let's talk about
BUGS

Fixing bugs is temporary. More bugs are coming.

Long-term impact comes from making bugs easier to fix, and eliminating or preventing classes of bugs.

A good refactor results in a long tail of bug fixing.

High-Return Technical Improvements:

- Code Access
- Build Process
- Testing Infrastructure and Automation
- Documentation
- Refactors that accomplish:
  - Major code reduction
  - Major improvements in internal compartmentation
  - Major tightening of internal APIs
  - Migration away from dangerous dependencies
- Bugs that are immediate security crises.

Information Security Practice Principles (ISPP)

- **Comprehensivity**: *Am I covering all of my bases?*
- **Opportunity**: *Am I taking advantage of my environment?*
- **Rigor**: *What is correct behavior, and how am I ensuring it?*
- **Minimization**: *Can this be a smaller target?*
- **Compartmentation**: *Is this made of distinct part with limited interactions?*
- **Fault Tolerance**: *What happens if this fails?*
- **Proportionality**: *Is this worth it?*

Finding Your Way In the Dark: Information Security From First Principles

5pm Wed

What makes devs' work harder?

High-Return Technical Improvements:

- Code Access
- Build Process
- Testing Infrastructure and Automation
- Documentation
- Refactors that accomplish:
  - Major code reduction
  - Major improvements in internal compartmentation
  - Major tightening of internal APIs
  - Migration away from dangerous dependencies
- Bugs that are immediate security crises.

What about interim maintenance?

Cyber-physical systems (ICS/SCADA/etc): emergency-fixing only...any other changes are at cross-purpose to the refactor.

With software, and with systems made up of general-purpose hardware components...you have a choice to make:

Emergency fixes only, focus all resources on the refactor.

--OR--

Develop in parallel: trade-off of lower end-user friction for MUCH higher resource and coordination needs during the rescue.

Build-and-Replace
vs.
Multi-Stage Refactor

Breaking it down: Priorities

Three themes must be considered at each step in planning and carrying out the refactor:

- **The Timeline Balancing Act:**
  Playing the long game while dealing with immediately pressing concerns, keeping perspective
- **Project Management Concerns:**
  Resourcing, communication, stakeholder priorities
- **Technical and Architectural Strategy:**
  Supporting toolchains, architecture principles, testing, code cleanliness, maintainability, security

- **Go for the snowball effect.**
- **Cluster disruptions in order to minimize them.**
- **Avoid Mythical Man-Month errors.**
- **Stay out of rabbit holes.**
- **Put long-term gains ahead of immediately pleasing people.**

The Cost of Continuous Time Estimation:

About 10% of your team's time.

Continuous Time Estimation with approval (assuming slowest response is 2-4 working hours):

15-20% of your team's time.

Q & A

Scenarios

Q & A

Many Thanks!

- To CTSC, the NSF Cybersecurity Center of Excellence, especially our director Von Welch, for the freedom to work on this, and to come teach it here.
- To OSG, for sending me to the Summit.
- To countless people who gave feedback along the way.

Open Science Grid

CTSC

CENTER FOR TRUSTWORTHY SCIENTIFIC CYBERINFRASTRUCTURE

The NSF Cybersecurity Center of Excellence

Don't stop now!

The slides and other info about this talk will be available via http://security.engineering/talks by tomorrow morning.

Come to "Finding Your Way In the Dark: Security From First Principles" tomorrow at 5pm to learn about how to secure your refactor, and more.

Reach me at sesons@iu.edu

Using and Sharing This Work: